Distributed Systems

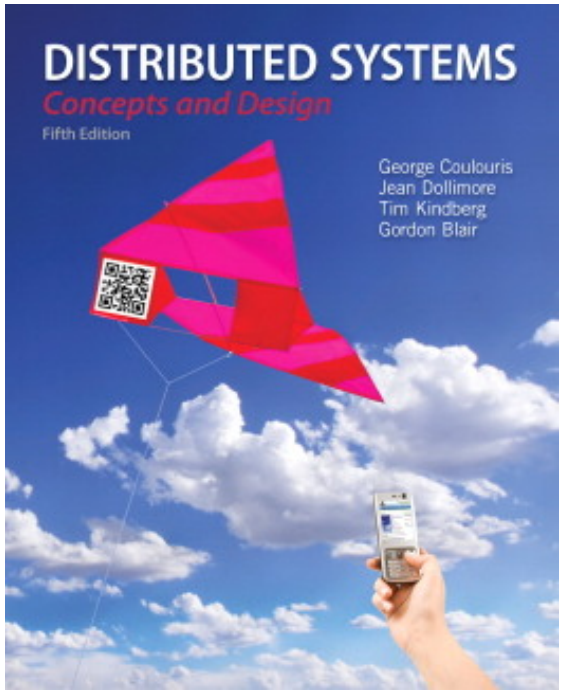
COMP90015 2019 SM1

Introduction

Lectures by Aaron Harwood

Textbook

From Coulouris, Dollimore and Kindberg, *Distributed Systems: Concepts and Design*, Edition 5, © Addison-Wesley 2012.



Materials used in all lecture slides with permission from Coulouris.

Search for and download an electronic copy (PDF) of the textbook online.

Staff

- Aaron Harwood, Lecturer, aharwood@unimelb.edu.au
- Irum Bukhari, Head Tutor, irum.bukhari@unimelb.edu.au
- Gayashan Amarasinghe, Tutor, gayashan.amarasinghe@unimelb.edu.au
- Oyemike Ebinum, Tutor, oyemike.ebinum@unimelb.edu.au
- Haowen Tang, Tutor, haowen.tang@unimelb.edu.au
- Qizhou Wang, Tutor, q.wang39@student.unimelb.edu.au
- Jiayun He, Tutor, jiayun.he@student.unimelb.edu.au

Assessment

- Project 1, 20%, software and written report (potentially multiple parts)
- Project 2, 20%, software and written report
- Final Exam, 60%, closed book, no calculator

All projects will be group work with target group size of 4. The language of choice is Java. Please start considering your project members now. Project details will be provided probably around Week 3 or 4. Computer labs around the university can be used to undertake programming work. We may schedule additional Java programming tutorials if there is enough demand; there is a forum thread for you to indicate your interest.

- Online Quiz Questions, not assessed, via LMS on a somewhat weekly basis

Academic Integrity

All University of Melbourne students are expected to uphold academic integrity in all aspects of every piece of work that they submit for assessment.

- Melbourne School of Engineering Academic Integrity
- Academic Integrity Module, Join the Community
- Academic Integrity Module, Re-visit the Community

Course Overview

- Introduction
- Models
- Interprocess Communication
- Remote Invocation
- Indirect Communication
- Operating System Support
- Security
- File Systems
- Naming

The order of the material may change, as well as the content. For this reason, lecture slides are not available all at the beginning, but rather weekly.

Introduction Overview

- What is a distributed system?
- Why distributed systems?
- Examples of distributed systems
- Challenges

What is a Distributed System?

Distributed System definitions - many and varying:

- A system in which hardware or software components located at networked computers communicate and coordinate their actions only by passing message [Coulouris]
- A collection of independent computers that appears to its users as a single coherent system [Tanenbaum]

Key aspects:

- a number of components
- communication between the components
- synergy; achieve more than the simple sum of individual components

Computer Networks vs Distributed Systems

A Computer Network: Is a collection of spatially separated, interconnected computers that exchange messages based on specific protocols. Computers are addressed by IP addresses.

A Distributed System: Multiple computers on the network working together as a system. The spatial separation of computers and communication aspects are hidden from users.

Why Distributed Systems?

- Resource Sharing:
  - Hardware Resources (Disks, printers, scanners etc)
  - Software Resources (Files, databases etc)
  - Other (Processing power, memory, bandwidth)
- Benefits of resource sharing:
  - Economy
  - Reliability
  - Availability
  - Scalability

Consequences of Distributed Systems

- Concurrency
- No global clock
- Independent failures

Computer Networks

Common distributed systems are based on widely used computer networks, e.g.:

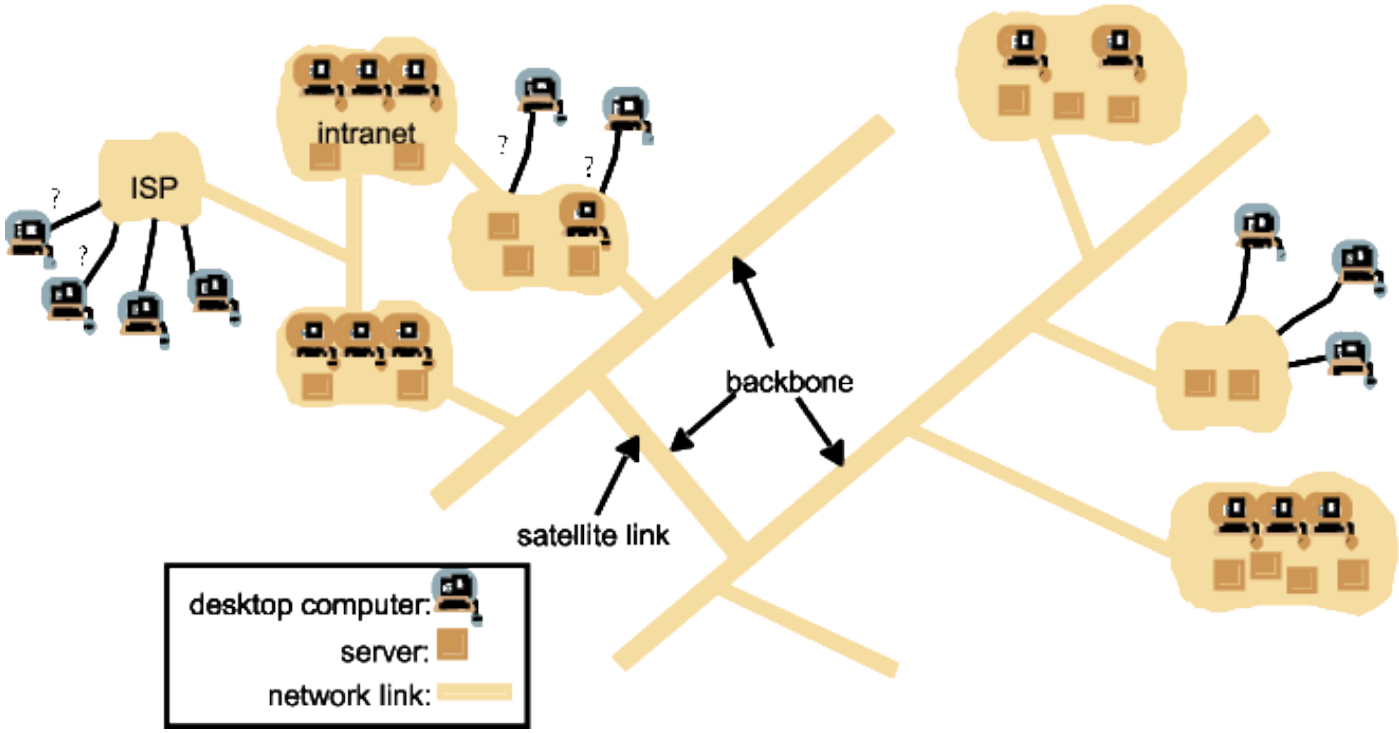
- The Internet
- Intranets
- Wireless networks

The Internet

Consists of a large number of interconnected collection of computer networks of different types. Features include:

- Computers interacting by message passing using a common means of communication (Internet protocol)
- Many different services (applications) (World Wide Web (www), email, file transfer)
- A number of Intranets linked by backbones
- Internet Service Providers (ISPs), that provide access to the services on the Internet while providing local service such as email and web hosting
- A backbone network link with high transmission capacity
- Communication via Satellite, fiber optic cables and other high-bandwidth circuits

A typical portion of the Internet

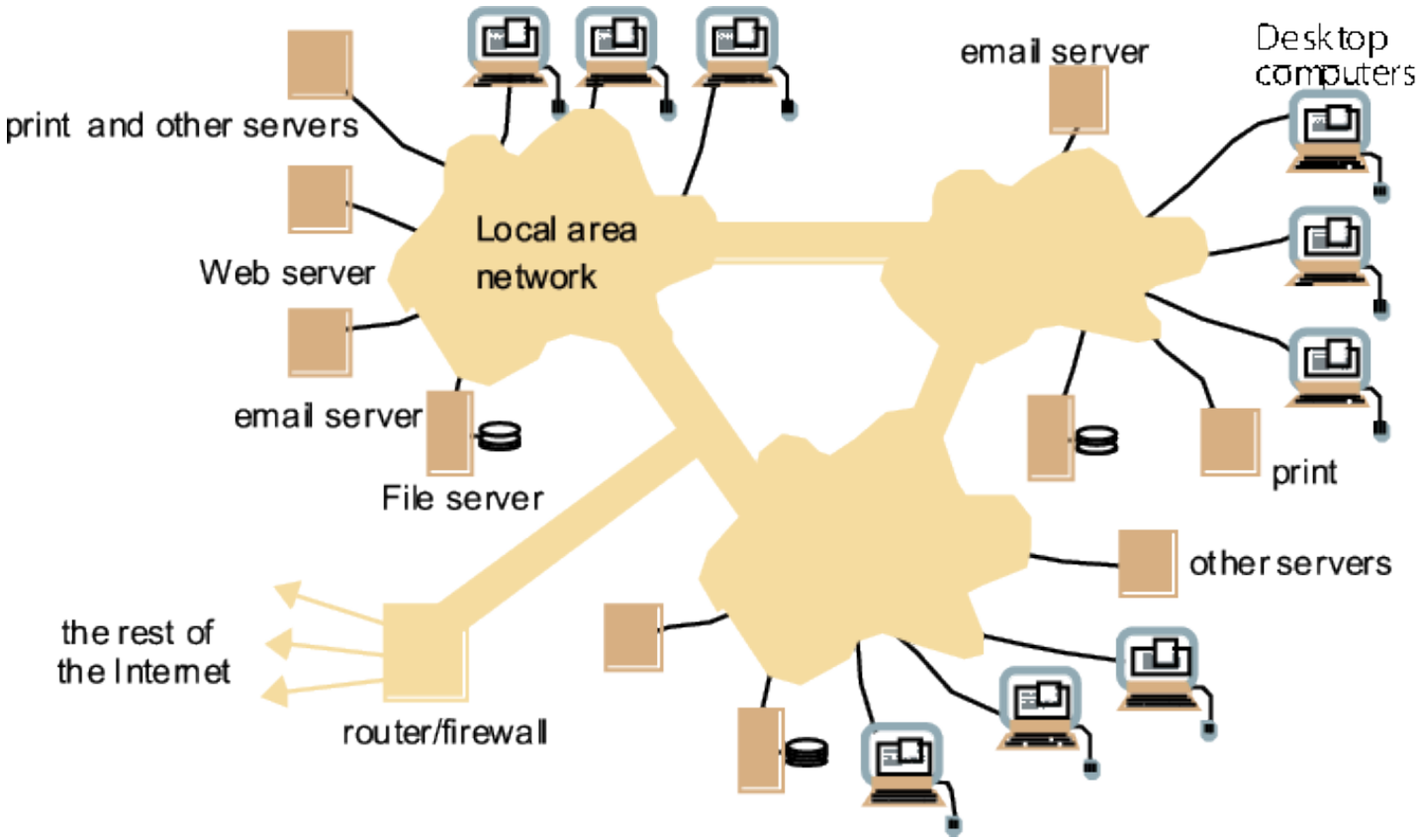


Intranets

Is a portion of the Internet that is separately administered by organizations. Features include:

- A boundary that can be configured to enforce local security policies
- Several local area connection (LANs) linked by backbone connections
- A connection to the Internet via a router allowing users within the intranet to access services on the Internet
- Firewalls to protect an intranet by preventing unauthorized messages leaving or entering by filtering incoming and outgoing messages e.g. by source or destination

A typical intranet



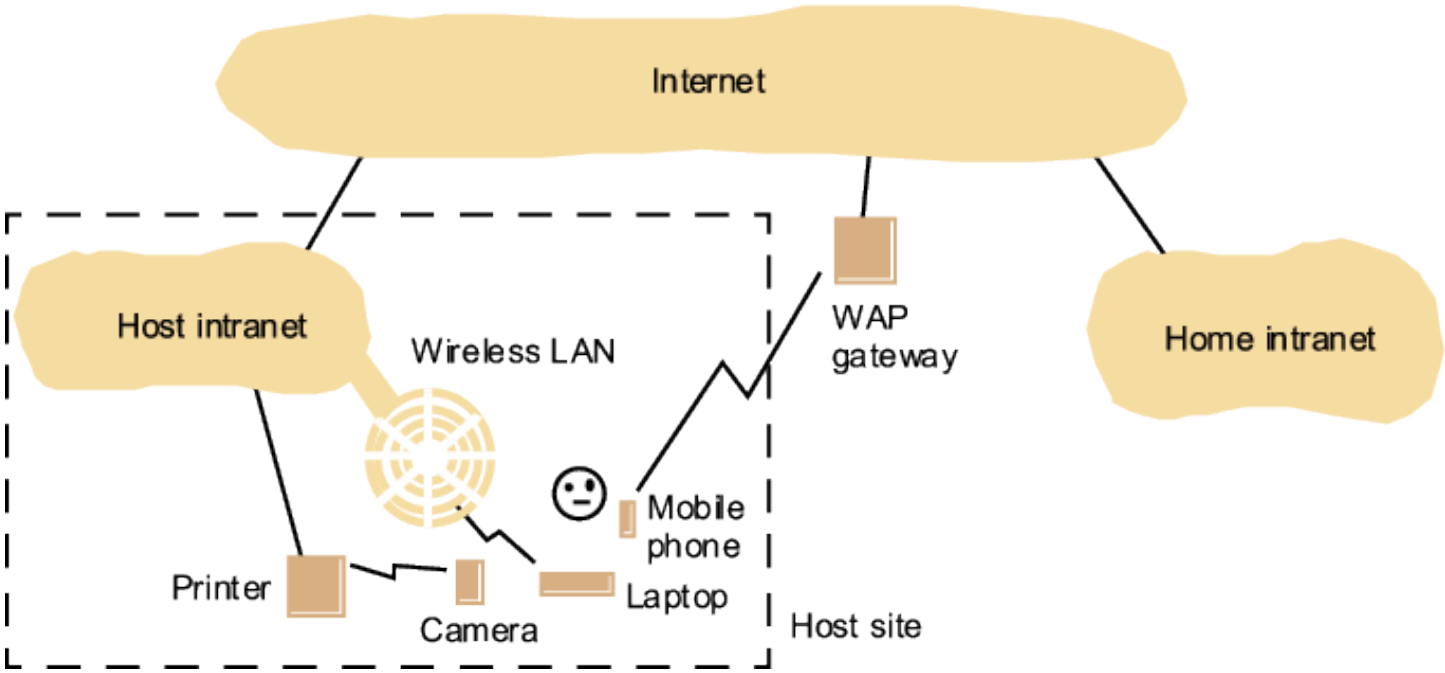
Wireless Networks

Wireless networking allows the integration of small and portable computing devices (e.g. laptop computers), hand-held devices (PDAs and mobile phones, video and digital cameras), wearable devices, device embedded applications (washing machines, refrigerators) into distributed systems.

Three popular (or not so popular) paradigms that may make heavy use of wireless networks are:

- Mobile Computing (Nomadic Computing)
- Ubiquitous Computing
- Internet of Things

Portable and hand-held devices in a distributed systems



Distributed system challenges

- Heterogeneity
- Openness
- Security
- Scalability
- Failure Handling
- Concurrency
- Transparency

Heterogeneity

Distributed systems use hardware and software resources of varying characteristics (heterogeneous resources):

- Networks
- Computer Hardware
- Operating Systems
- Programming Languages
- Implementation by different developers

Some approaches for handling heterogeneity issues are:

- Using standard protocols
- Using agreed upon message formats and data types
- Adhering to an agreed upon Application Program Interfaces (APIs)
- Using Middleware
- Portable code

Middleware is a software layer between the distributed application and the operating systems that:

- provides a programming abstraction and
- masks the heterogeneity of the underlying networks, hardware, operating systems and programming languages, e.g. RMI.

Certain middleware solutions address some heterogeneity issues but not others.

Middleware Models:

- Distributed File Systems
- Remote Procedure Call (RPC) (procedural languages)
- Remote Method Invocation (RMI) (object-oriented languages)
- Distributed Documents
- Distributed Data Bases

Heterogeneity and mobile code

Mobile code is sent from one computer to the anther to run at the destination (e.g. Java applets):

- Code that is compiled to run in one OS does not run in another:
  - different hardware
  - different OS versions
- Virtual Machine approach provides a way of making code executable on any hardware - compiler produces code that is interpreted by the virtual machine
- Cross-platform compilation and code portability is another way, that compiles source code to multiple targets.

Openness

Openness refers to the ability of extend the system in different ways by adding hardware or software resources. Following are some approaches to address openness:

- Publishing key interfaces
- Allowing a uniform communication mechanism to communicate over the published interfaces
- Ensuring all implementations adhere to the published standards

Security

- There has three aspects of security:
  - Confidentiality (protection against disclosure to unauthorized individuals)
  - Integrity (protection against alteration and corruption)
  - Availability (protection against interference with the means of access)
- Security Mechanisms:
  - Encryption (e.g. Blowfish, RSA)
  - Authentication (e.g. passwords, public key authentication)
  - Authorization (e.g. access control lists)

Security

Types of security challenges have not yet been resolved completely:

- Denial of service attacks
- Security against mobile code (executables as attachments)

Scalability

A system is considered to be scalable if it can handle the growth of the number of users. Scalability Challenges:

- Cost of physical resources: For a system with n users to be scalable, the quantity of physical resources required to support the system should be O(n) - if one file server can support 20 users, then two servers should be able to support 40 users.
- Controlling the performance loss: Complexity of algorithms that are used for searches, lookup etc should be scalable - for example a search algorithm for n data items requiring O(log(n)) search steps is scalable but one that requires n^2 is not.
- Resources should not run out: e.g. 32-bit Internet IP addresses running out requiring a new version of 128-bit address.
- Avoiding Performance bottlenecks: Decentralized architectures and algorithms can be used to avoid performance bottlenecks.

Failure Handling

Following are approaches to handing failures:

- **Detecting:** some types of failures can be detected, e.g. checksums can be used to detect corrupted data in a message, and some kinds of failure are hard to be certain about, e.g. the failure of a remote server.
- **Masking:** some failures that have been detected can be hidden or made less severe, e.g. timeout and message retransmission.
- **Tolerating:** it is sometimes impractical to try and handle every failure that occurs, sometimes it is better to tolerate them, e.g. failure is reported back to the user, e.g. web server not being available, try again later, or video is rendered with errors
- **Recovery:** failure sometimes leads to corrupted data and software can be designed so that it can recover the original state after failure, e.g. implementing a roll back mechanism.
- **Redundancy:** services can be made to tolerate failures using redundant components, e.g. multiple servers that provide the same service, as so called fail over.

Concurrency

- Multiple clients can access the same resource at the same time, in some cases for updates
- One approach to handling concurrency is making access sequential - slows down the system
- Semaphores supported by the operating system is a well accepted mechanism to handle concurrency

Transparency

This is the aspect of hiding the components of a distributed system from the user and the application programmer. Types of transparencies:

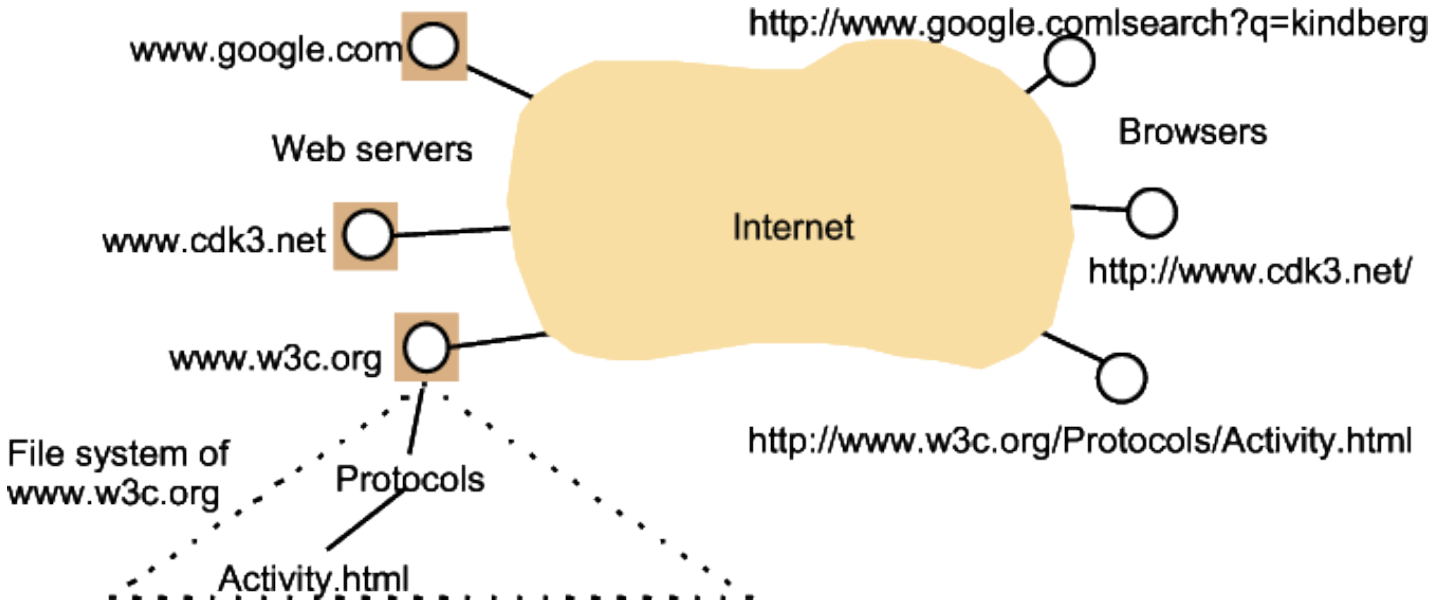
- Access transparency
- Location transparency
- Concurrency transparency
- Replication transparency
- Failure transparency
- Mobility transparency
- Performance transparency
- Scaling transparency

The World Wide Web (example)

Began as a simple system for exchanging documents at the European centre for nuclear research (CERN), Switzerland, in 1989. Documents are logically organized using a *hypertext* structure and are connected through *links*. The three main standard technological components used in the web are:

- HypterText Markup Language (HTML) - This is a language for specifying the contents and layout of pages to be displayed by browsers.
- Uniform Resource Locators (URLs) - These identify the resources stored on the web. A URL has two components: *scheme* and *scheme-specific-identifier*. Scheme - declares the type of resource. Scheme-specific-identifier - identifies the resource. E.g., *mailto:xxx@yahoo.com*, *ftp://ftp.downloadIt.com/aProg.exe*
- HyperText Transfer Protocol (HTTP) - This is the protocol used for transferring resources between web servers and clients (e.g. browsers). The main features are:
  - Uses request-reply interactions
  - Supports different content types
  - Requires one request per resource
  - Supports simple access control

Web servers and web browsers



Dynamic pages

- Static web pages allow data to be made available for retrieval
- Dynamic web pages allow users to interact with resources by taking user input, executing programs and returning results
- Programs executed on the request can take different forms:
  - A Common Gateway Program (CGI)
  - Servlet
  - Downloaded code - Javascript, applet

Web Services & APIs

- Web services paradigm allows programs (not just browsers) to be clients for web programs
- Data between the server and the client is passed using Extensible Markup Language (XML).
- SOAP protocol allows client to invoke web services.

Growth of APIs available to provide various services:

- Google (searching and maps)
- Facebook (social networking)
- Twitter (information)
- Amazon (marketplace)
- YouTube (video sharing)
- Flickr (photo sharing)
- Dropbox (file/document sharing)

Summary

- Distributed systems are common
- Communication networks that enable distributed systems are the Internet, intranets and wireless networks
- Resource sharing is the main motivation for distributed systems
- There are many challenges associated with these systems - Heterogeneity, Openness, Security, Scalability, Failure handling, Concurrency, Transparency